Python Version used - python 3

1.

A brief discussion of how you have implemented the LSR protocol. Provide a list of features that you have successfully implemented. In case you have not been able to get certain features of LSR working, you should also mention that in your report.

A router reads from the configuration file (configA.txt for example) provided to it. This file has information of the name of the router, the port it has to listen on, number of neighbours, name of neighbours, cost to reach them and the ports they would listen on. This information is essentially its Link State. Every router sends 2 types of messages to its neighbours:-

a. LSA- A router will advertise its link state to its neighbours. Since the message could be sent from any port (provided at random by OS), it will have itself as the 'sender' and 'time' at which it sends the message along with the link state to its neighbours. This will be done every second.

b. Forward LSA- A router will continuously listen for and receive LSA(of other routers) which it needs to forward to its neighbours. A router maintains information about the last time it received LSA of any router. Comparing this time with the timestamp in the message it will see if the message received is a new one or a stale one. If it is new, then it will put itself as the 'forwarder' and forward it to its neighbours else don't process further.

After sometime, it will have an idea about the global topology and implement Dijkstra's algorithm to find the least cost path to other routers. This happens every 30 seconds.

2.

Describe the data structure used to represent the network topology and the link-state packet format. Comment on how your program deals with node failures and restricts excessive link-state broadcasts.

The network topology is maintained as a dictionary of dictionary (nested dictionaries) like - {'F': {'A': 2.2, 'D': 0.7, 'E': 6.2}, 'B': {'A': 6.5, 'C': 1.1, 'D': 4.2, 'E': 3.2}, 'E': {'B':

3.2, 'D': 2.9, 'F': 6.2}, 'A': {'B': 6.5, 'F': 2.2}, 'D': {'F': 0.7, 'B': 4.2, 'C': 1.6, 'E': 2.9}, 'C': {'B': 1.1, 'D': 1.6}}

It follows as router as key and its neighbours and cost to reach in a dictionary as the value. Link state packet is of the form -

{‘neighbours': {'B': 6.5, 'F': 2.2}, 'sender': 'A', 'time': 1564900562.152833}

This dictionary will also have a ‘forwarder’ field if a router is not the creator of the message but has to forward it.

The program periodically checks for dead routers. They are those routers whose LSA it hasn’t received in some time(3-4s). This can be checked by seeing the time difference between the current time and the time it maintains. If the maintained time is older by 3-4s (for neighbours) or 13-14s (for distant routers), it means that the router is dead. This router then shall be removed from its neighbours and the global topology.

Link state broadcasts are being limited as:-

a. If the timestamp in the message is older than the timestamp the router maintains for all the routers of the topology then it will consider this message as old and won’t process it further.

b. If the message is found to be new then the router will try to forward it. However, it will only forward to those neighbours that are neither the original sender nor the forwarder(both fields maintained and present in the message).

3.

Discuss any design trade-offs considered and made. List what you consider is special about your implementation. Describe possible improvements and extensions to your program and indicate how you could realise them.

a. String processing is involved but dict() is a very convenient and flexible data structure.

b. Every message is decoded and needs to be converted as bytes->string->dictionary.

Despite the processing, the implementation remains special in the fact that :-

a. LSA message is always constructed from the current neighbours. Change in topology is almost instantly reflected in the next Link State Advertisement.

b. The main content of the message that is the neighbours information is sent in such a way that it can be directly used as is to form the global topology.

c. Time check for dead neighbours has been taken as (3-4s) and for distant neighbours as (13s) which takes into consideration that there will be only 10 nodes and situated in worst possible way - straight line.

Possible Improvements :-

a. Number of LS broadcasts could be further limited by not broadcasting to the common neighbours of the router and sender and router and forwarder. But this assumes that messages from the sender and forwarder have correctly reached and processed by their respective neighbours. We cant know that. This implies redundant transmissions, but it won't have much impact as node will have already received this message and see it as a stale message.

4.

Indicate any segments of code that you have borrowed from the Web or other books.

The code to calculate Dijkstra's least cost path (def dijkstra(nodesDict: dict, current: str)) has been borrowed from -

https://stackoverflow.com/a/22899400/4933540

However, it was only calculating the least distance and not the path it takes along the way. Appropriate modifications have been made to accomodate that.